✓Why we need OOP

✓Procedural languages/Structured Programming

✓Problems with Structured Programming

✓Object Oriented Approach

Motivation for OOP

For Detailed Reading, Consult

Object-Oriented Programming *by* ***Robert Lafore****, 4th Ed. Chatper-1*

Why We Need OOP

Object-oriented programming was developed because limitations were discovered in earlier approaches to programming

Structured Programming
Procedural Languages

C, Pascal, FORTRAN, and similar languages are *procedural languages*.

- For very small programs
  - No other organizing principle (often called a *paradigm*) is needed
- For large programs
  - Division into functions

Dividing a program into functions and modules is one of the cornerstones of *structured programming*

Structured Programming
Problems with Structured Programming

- There are two related problems
  - Functions have **unrestricted access** to global data
  - **Unrelated functions and data**, provide a poor model of the real world.

Structured Programming

Problems with Structured Programming

Unrestricted Access

- In a procedural program, there are two kinds of data
  - ***Local data***
  - *Global data*

    - hidden inside a function
    - is used exclusively by the function
    - safe from modification by other functions

Structured Programming

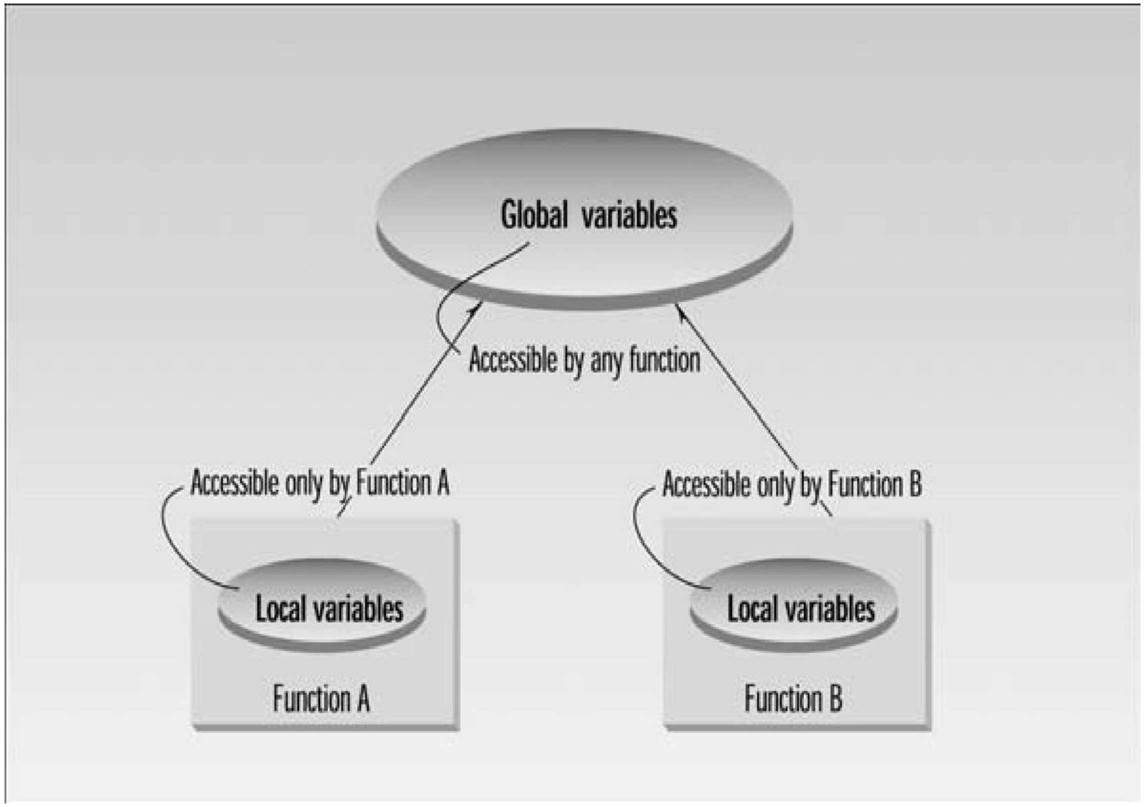
Problems with Structured Programming

Unrestricted Access

- In a procedural program, there are two kinds of data
  - *Local data*
  - ***Global data***
    - When two or more functions must access the same data then the data must be made *global*
    - Global data can be accessed by *any* function in the program

Structured Programming
Problems with Structured Programming
Unrestricted Access

- In a procedural program, there are two kinds of data
  - ***Local data***
  - ***Global data***

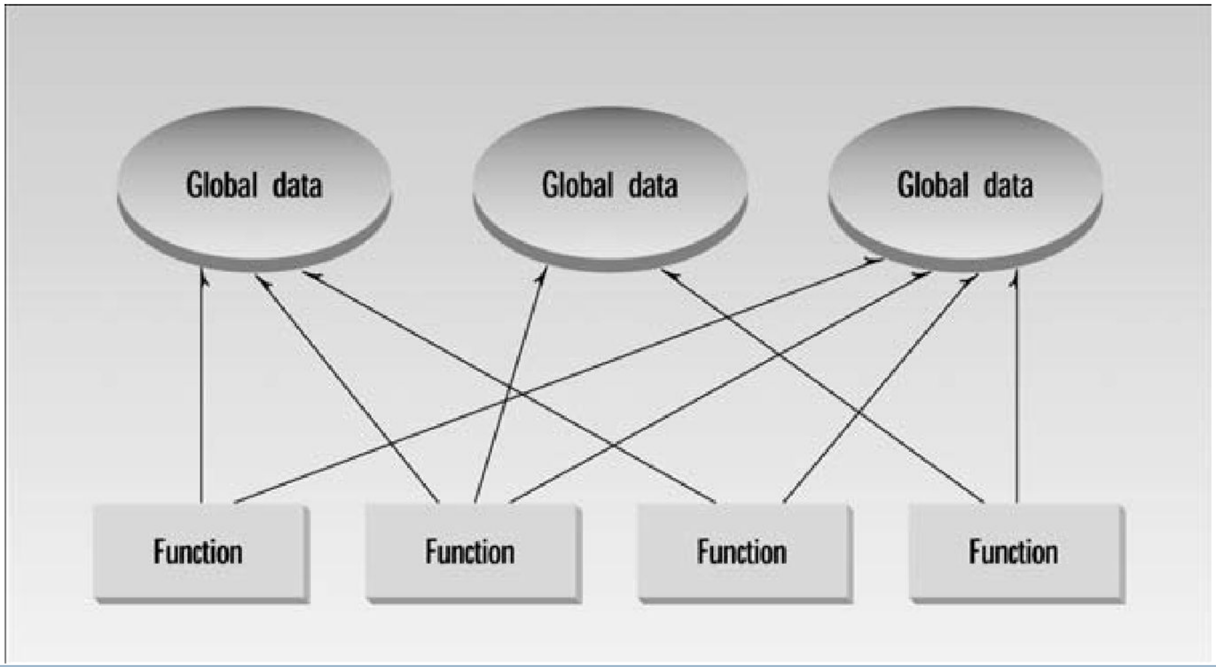

Global variables
Accessible by any function
Accessible only by Function A
Accessible only by Function B
Local variables
Function A
Local variables
Function B

Structured Programming

Problems with Structured Programming

Unrestricted Access

In a large program, there are many functions and many global data items. The problem with the procedural paradigm is that this leads to an even larger number of potential connections between functions and data



Structured Programming

Problems with Structured Programming

Unrelated Functions and Data

- Arrangement of separate data and functions does a poor job of modeling things in the real world
- In the physical world we deal with **objects** such as people and cars
- Objects aren’t like data and they aren’t like functions
- Real-world objects have both *attributes(Data)* and *behavior(Functions)*.

Modeling the Real world

- Real world objects has
  - **Attributes**
  - Behavior

Examples of attributes (sometimes called *characteristics*) are:

- For people
  - Eye color
  - Job title
- For Cars
  - Horsepower
  - Number of doors

Attributes in the real world are equivalent to data in a program

Modeling the Real world

- Real world objects has
  - Attributes
  - **Behavior**

Behavior is something a real-world object does in response to some stimulus:

- If you ask your boss for a raise, she will generally say yes or no
- If you apply the brakes in a car, it will generally stop

Behavior is like a function: you call a function to do something and it does it

So neither data nor functions, by themselves, model real-world objects effectively

The Object Oriented Approach

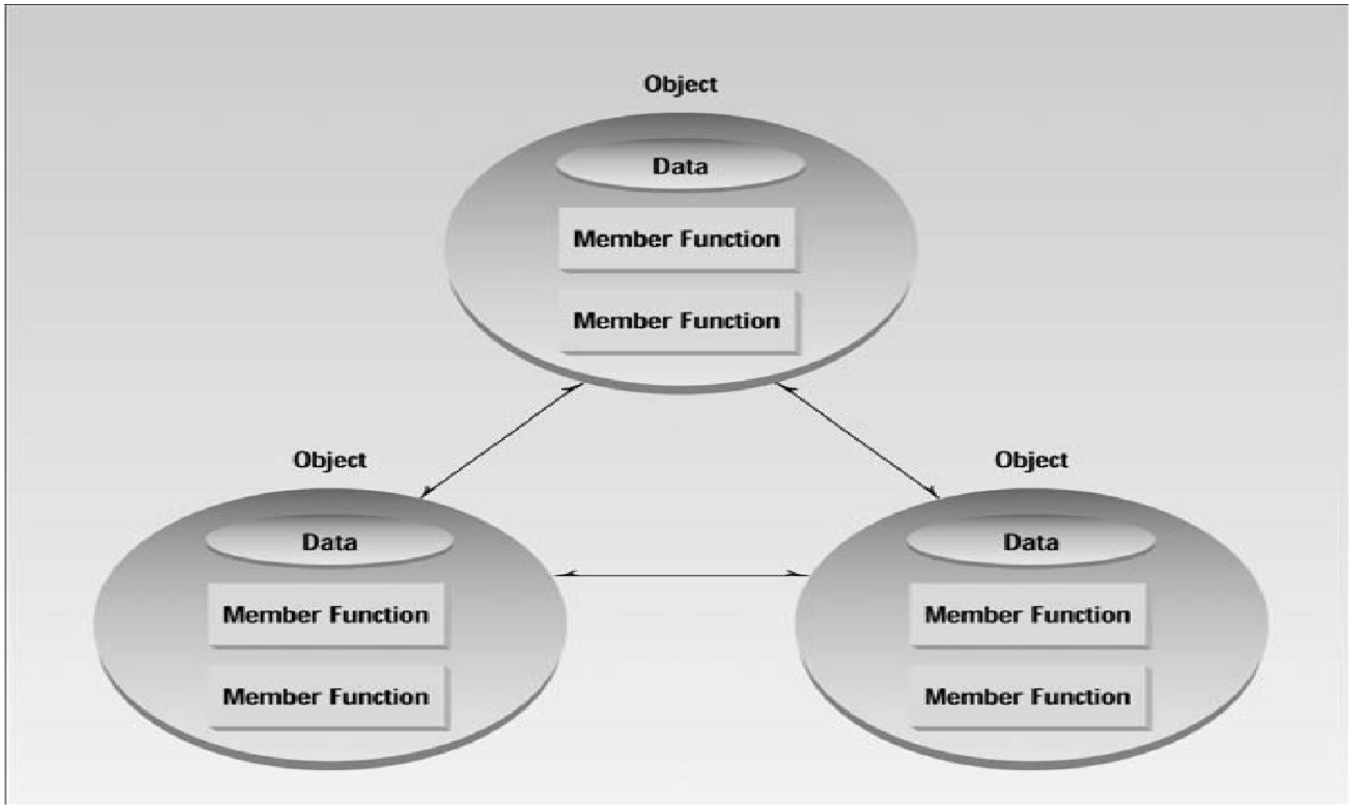
- The fundamental idea behind object-oriented languages is to combine into a single unit both *data* and the *functions that operate on that data*.
- Such a unit is called an *object*.

- Data and its functions are said to be *encapsulated* into a single entity

- An object's functions, called *member functions* in C++, typically provide the only way to access its data.
- You can't access the object's data, called *attributes* or *instance variables,* directly. The data is *hidden*, so it is safe from accidental alteration

The Object Oriented Approach

A C++ program typically consists of a number of objects, which communicate with each other by calling one another's member functions


Object
Data
Member Function
Member Function
Object
Data
Member Function
Member Function
Object
Data
Member Function
Member Function